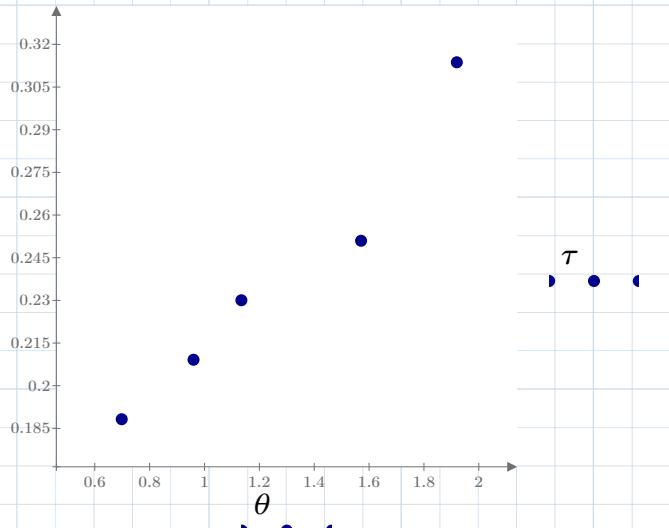


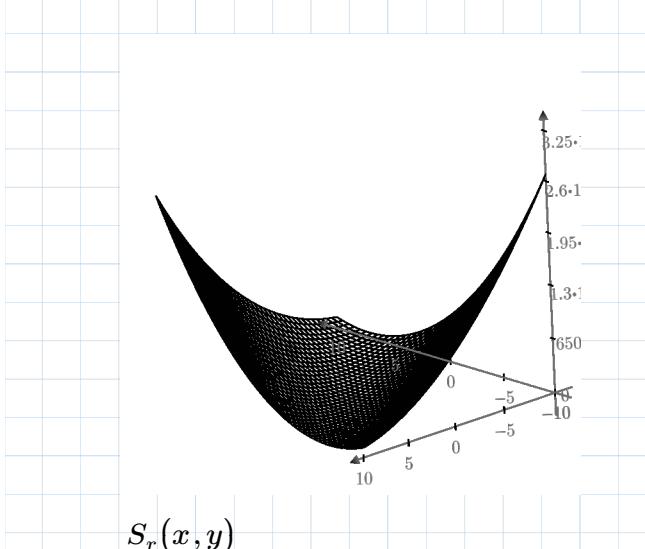
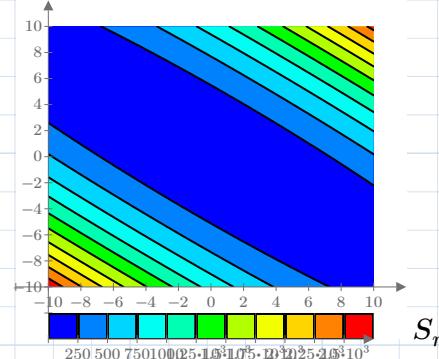
θ	τ	
0.698132	0.188224	To the left is a dataset from Kau's NM-MFC 06.03's Example 1 where we are playing with mousetraps...
0.959931	0.209138	
1.134464	0.230052	Let's use this case to explore basic linear regression theory and execution. We can also take this example to illustrate how linear regression, correlation and predictive skill are related.
1.570796	0.250965	
1.919862	0.313707	
First, let's plot these two fields against each other...		
It clearly makes a nice linear relationship.		
Let's now pull some basic stats...		
$\theta_{mean} := \text{mean}(\theta) = 1.257$ $s_\theta := \text{Stdev}(\theta) = 0.488$ $\tau_{mean} := \text{mean}(\tau) = 0.238$ $s_\tau := \text{Stdev}(\tau) = 0.048$ $N := \text{rows}(\theta) = 5$ $\alpha := 100\% - 95\% = 0.05$		
$CI95_\theta := \text{qt}\left(1 - \frac{\alpha}{2}, N - 1\right) \cdot \frac{s_\theta}{\sqrt{N}} = 0.606$ $CI95_\tau := \text{qt}\left(1 - \frac{\alpha}{2}, N - 1\right) \cdot \frac{s_\tau}{\sqrt{N}} = 0.06$		 <p>We will be creating a function to fit the dependent variable τ to independent variable θ</p> $\tau_\theta(\theta) = a_0 + a_1 \cdot \theta$
Remember the secret to linear regression. We want to minimize the error between our regression equation results and our original values of tau. We express it as a summed square of the residuals (SSR) or S_r . We can also express this as a "Cost" or "Error" function if we were to look at this problem from a very general sense.		
$S_r(a_0, a_1) := \sum \left((a_0 + a_1 \cdot \theta) - \tau \right)^2$		
If we were to take our values of tau and theta up top and plug them into the above equation (and shrink the font down to microscopic scale so we can see the whole thing) we get the following polynomial function for which we now must solve for a_0 and a_1		
$S_r(a_0, a_1) \xrightarrow{\text{simplify}} 5.0 \cdot a_0^2 + 12.56637 \cdot a_0 \cdot a_1 - 2.384172 \cdot a_0 + 8.849134554141 \cdot a_1^2 - 3.179275851496 \cdot a_1 + 0.293486412998$		
Put in other words we need to calibrate our values of a_0 (y-intercept) and a_1 (slope) to minimize the error that is crunched by the formula above.		
So how do we do this in a way that is efficient, accurate and mathematically sound?		

$$S_r(a_0, a_1) \xrightarrow{\text{simplify}} 5.0 \cdot a_0^2 + 12.56637 \cdot a_0 \cdot a_1 - 2.384172 \cdot a_0 + 8.849134554141 \cdot a_1^2 - 3.179275851496 \cdot a_1 + 0.293486412998$$

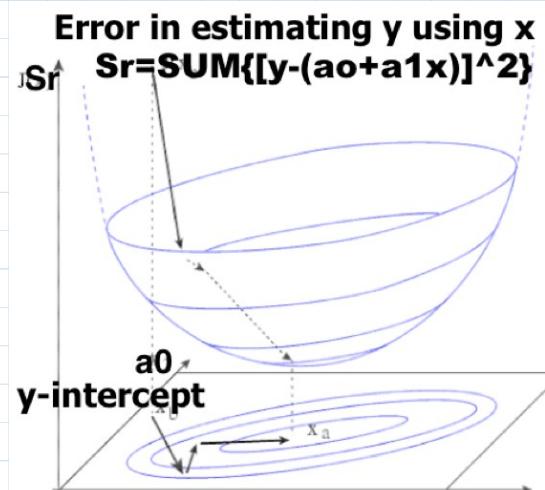
Here we are plotting S_r vs a_0 and a_1 in a contour plot

Somewhere is the blue is the minimum S_r .

Below is it plotted in 3D and to the side is an idealized version our error function.



x:a0, y-intercept
y:a1, slope
z: S_r /SSR/Error/Cost...



from Bouttier and Courtier 1999
An idealization of a 2-D representation

If you remember your Calc-Fu there is a very straight forward to to do this. Our magic answer for our slope and intercept will be where the gradient (derivative) of our SSR/ S_r /Cost function is zero.

This method not only applies to linear regression but also to multivariate regression and many other types of calibration. Including those cases where we are calibrarting not just two parameters like slope and intercept, but 10s or even 100s of parameters.

So let's take our S_r function and take the a_0 (intercept) and a_1 (slope) derivative of it.

$$S_r(a_0, a_1) \xrightarrow{\text{simplify}} 5.0 \cdot a_0^2 + 12.56637 \cdot a_0 \cdot a_1 - 2.384172 \cdot a_0 + 8.849134554141 \cdot a_1^2 - 3.179275851496 \cdot a_1 + 0.293486412998$$

So let's take our S_r function and take the a_0 (intercept) and a_1 (slope) derivative of it.

$$\frac{d}{da_0} S_r(a_0, a_1) \rightarrow 10 \cdot a_0 + 12.56637 \cdot a_1 - 2.384172$$

$$\frac{d}{da_1} S_r(a_0, a_1) \rightarrow 12.56637 \cdot a_0 + 17.698269108282 \cdot a_1 - 3.179275851496$$

So for our problem, our a_0 (intercept) and a_1 (slope) will be valid where both derivatives/slopes [or together, the "gradient"] are equal to zero.

$$\begin{bmatrix} \frac{\partial S_r(a_0, a_1)}{\partial a_0} \\ \frac{\partial S_r(a_0, a_1)}{\partial a_1} \end{bmatrix} = 0$$

Or if you've had vector Calc (in Calc 2 or Calc 3 depending on your college)

$$\vec{\nabla} S_r(\vec{a}) = 0$$

Where the vector a represents all of the coefficients that you need to calibrate whether it's just

slope & intercept,

"slopes" for cement and water & "intercept"
1000 variables' "slopes" & "intercept"

With respect to our problem, see just have to
solve a_0 and a_1 for these two equations.

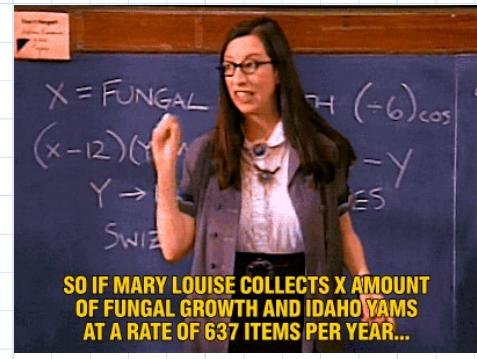
$$10 \cdot a_0 + 12.56637 \cdot a_1 - 2.384172 = 0$$

$$12.56637 \cdot a_0 + 17.698269108282 \cdot a_1 - 3.179275851496 = 0$$

So now our seemingly scary problem becomes an even scarier flashback to Junior High Algebra class with Mrs. Mercer.

This also becomes a blend of classic "root" or "zero" problem and linear algebra problem in Numerical Methods.

Either way, Mathcad actually has several ways to attack this problem.



For starters, let's just cut to the easy answers. There are established formulas for getting the slope and intercept for a two-variable calibration (which is what linear regression is)

$$slope_c := \frac{N \cdot \sum_{i=0}^{N-1} (\theta_i \tau_i) - \sum \theta \cdot \sum \tau}{N \cdot \sum \theta^2 - \left(\sum \theta \right)^2} = 0.096 \quad yint_c := \tau_{mean} - slope_c \cdot \theta_{mean} = 0.118$$

And with so many other frequently used operations, there are intrinsic functions to do this

$$A_1 := \text{slope}(\theta, \tau) = 0.096$$

$$A_0 := \text{intercept}(\theta, \tau) = 0.118$$

So far so good. Everything Matches!

But since this Sandbox is more than about just plugging and chugging.
Let's see what other tricks in Matchad we can use that may be useful in other problems

If you haven't been using units, the Maximum-Satisfaction-With-Minimal-Effort Tactic is to use the Sybolic Arrow with the Solve Command

$$\left[\begin{array}{l} \frac{d}{da_0} S_r(a_0, a_1) \\ \frac{d}{da_1} S_r(a_0, a_1) \end{array} \right] \xrightarrow{\text{solve, } a_0, a_1} [0.11766514898834059588 \ 0.096091433732779954845] \quad \begin{array}{l} a_0: \text{Slope} \\ a_1: \text{Slope} \end{array}$$

Again we have a successful bench. All values are in agreement.

Here's another function that extracts the local minimum value of any function. (Mathcad has one for the maximum value too)

$a_0 := 0 \quad a_1 := 0$ You need arbitrary guess values for a_0 and a_1 for this one...

$$\text{minimize} (S_r, a_0, a_1) = \begin{bmatrix} 0.118 \\ 0.096 \end{bmatrix} \quad \text{Still doing great!}$$

And finally a structure in Mathcad alled a "Solve Block" which is rather popular with students since it can do a lot of different things.

And with all of these methods, we get the same values!

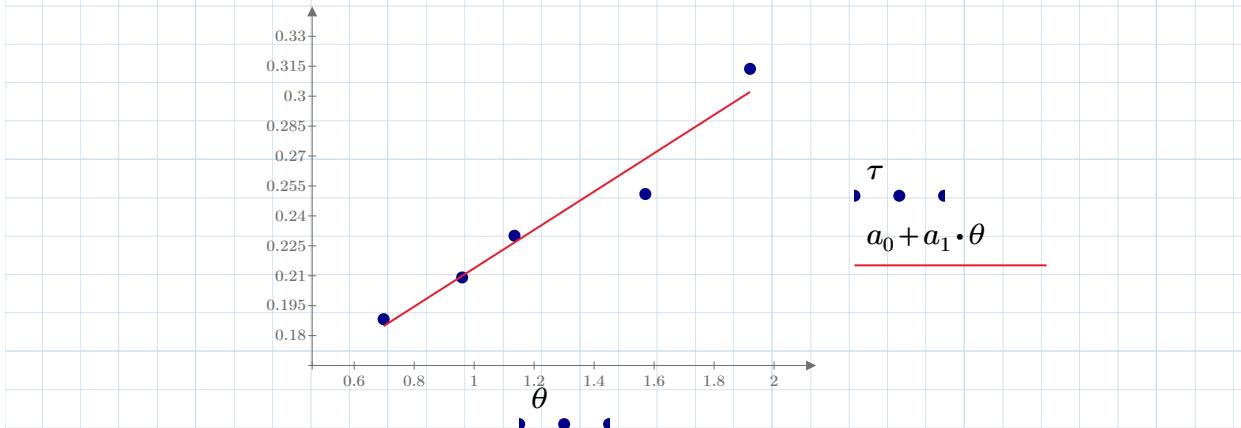
Guess Values	$a_0 := 0$	$a_1 := 0$
Constraints	$\frac{d}{da_0} S_r(a_0, a_1) = 0$	$\frac{d}{da_1} S_r(a_0, a_1) = 0$
Solver	$\text{find} (a_0, a_1) = \begin{bmatrix} 0.118 \\ 0.096 \end{bmatrix}$	

Now let's examine the quality of our regression.

Let's start by creating a new formula for our linear regression

$$a_0 := \text{intercept}(\theta, \tau) = 0.118 \quad a_1 := \text{slope}(\theta, \tau) = 0.096 \quad \tau_\theta(\theta) := a_0 + a_1 \cdot \theta$$

And it's always a good idea to plot things out....



Just by visual inspection you should expect a low RMSE and high correlations from the look-n-feel of this graph.

It makes a strong line through the data (with just one outlier that stands out)

The Total Sum of Squares (St or SST) is the baseline **unskilled** error and is the unskilled analog (i.e., equivalent) to the "**skilled**" Sum of the Squared Residuals (Sr or SSR). This is basically the variance formula of your dependant variable (torque) without dividing by N

$$S_t := \sum \left((\tau - \tau_{\text{mean}})^2 \right) = 9.273 \cdot 10^{-3}$$

And our Sr
(our **skilled** error) is...

$$S_r := \sum \left((\tau - \tau_\theta(\theta))^2 \right) = 4.688 \cdot 10^{-4}$$

Clearly we have an improvement here since Sr is notably less than St.

So let's articulate it with our other error metrics.

Our RMSE and Standard Error of the Estimate are...

$$RMSE := \sqrt{\frac{S_r}{N}} = 9.683 \cdot 10^{-3} \quad s_{\tau_\theta} := \sqrt{\frac{S_r}{N-2}} = 1.25 \cdot 10^{-2}$$

And let's now create a basic baseline measure for skill where the error for a perfect forecast skill is zero. Remember that a good working definition of "*skill*" is how much better your method is to random chance or other *unskilled* method

$$\text{Skill} = \frac{\text{yours} - \text{unskilled}}{\text{perfect} - \text{unskilled}} \quad \text{skill} := \frac{S_r - S_t}{0 - S_t} = 0.949 \quad r2 := \frac{S_t - S_r}{S_t - 0} = 0.949$$

And tying it to previous horseplay...

$$\text{corr}(\theta, \tau) = 0.974 \quad \text{corr}(\theta, \tau)^2 = 0.949$$

So we now can officially link correlation with predictive skill!

Now let's explore how to integrate confidence limits into our plots.

The formula for a 95% confidence limit is here:

$$\alpha := 100\% - 95\% = 0.05$$

$$CI95(\theta) := qt\left(1 - \frac{\alpha}{2}, N - 2\right) \cdot s_{\tau\theta} \cdot \sqrt{\frac{1}{N} + \frac{(\theta - \theta_{mean})^2}{(N - 1) \cdot s_{\theta}^2}}$$

Notice in the above formula I have two flavors of "theta". The lowercase thetas are referring to the thetas in our original dataset. I am using the capital Theta for a "function variable" for any value of theta we chose to throw at it.

Now for the Prediction Interval - remember. You really should be wary of using this for original datasets that are not normally distributed. While we can do symmetric CIs for the means thanks to the Central Limit Theorem we cannot do so for original data that is not a nice bell-curve normal distribution.

$$PI95(\theta) := qt\left(1 - \frac{\alpha}{2}, N - 2\right) \cdot s_{\tau\theta} \cdot \sqrt{1 + \frac{1}{N} + \frac{(\theta - \theta_{mean})^2}{(N - 1) \cdot s_{\theta}^2}}$$

Let's all recall the structure of Confidence Limits by exploring math in the functions above.

We should have more certainty for a linear regression when we are in the center of the known data. As a result the confidence limits should be the thinnest there.

As we go outward into "here be dragons" territory, we will want to spread out those intervals. Likewise as we have greater net errors between our regression and our individual data points our CI (and PI) should once again fatten. If an "x" axis independent variable used to create the regression has a lower spread, we will also see wider CI thicknesses as you approach terra incognito.

Finally the more data we have to make a regression (or any other type of estimate) should also carry higher confidence and this make the regression confidence limits skinny.

And when plotted out, we get the chart to the right (remember that the Prediction Limits shown here are mostly demonstrative. Close inspection of the stats shows that we are NOT looking at normally-distributed data)

